Supporting Online Material

Methods

Network equations

The ESNs described in the article are standard discrete-time sigmoid networks. The state update equation is $\mathbf{x}(n+1) = \tanh(\mathbf{W}\mathbf{x}(n) + \mathbf{w}_{in}\,u(n+1) + \mathbf{w}_{fb}\,y(n) + \nu(n))$ [W: $N \times N$ matrix of internal connection weights; $\mathbf{w}_{in}$: N-size vector of input connection weights; $\mathbf{w}_{fb}$ (optional): weight vector for feedback connections from the output neuron to the reservoir; $\nu(n)$ (optional): noise vector; the tanh is applied elementwise]. The output equation for a single-output network is $y(n) = \tanh(\mathbf{w}_{out}(\mathbf{x}(n), u(n)))$ [$\mathbf{w}_{out}$: (N+1)-size vector of weights of connections to the output neuron; $\mathbf{w}_{out}$ contains the w_i used in the article text]. The output nonlinearity tanh is optional. In the Mackey-Glass experiment it was present, whereas in the channel equalization study we used linear output units. In order to make a random reservoir qualify as an echo state network, it must exhibit certain damping properties. They can be ascertained by ensuring that the weight matrix **W** has a spectral radius smaller than unity (*S1*).

The Mackey-Glass system

Preparation of training and testing data. The MG delay differential equation $dx/dt = 0.2\,x(t-\tau)/(1 + x(t-\tau)^{10} - 0.1\,x(t))$ [τ: delay; here we used $\tau = 17$, the value standardly employed in most of the MGS prediction literature] was simulated using the `dde23` solver for delay differential equations from the commercial toolbox Matlab. This solver allows one to specify the absolute accuracy; it was set to 1e-16. A stepsize of 1.0 was used (which is finer than the stepsize automatically selected by `dde23`). The resulting time series were shifted by –1 and passed through a tanh function, whereby they fell into a range of (–0.5, 0.3). From all data series thus generated, the first 1000 steps were discarded to get rid of initial washouts.

Network setup. To construct the ESN, a 1000 x 1000 reservoir weight matrix **W** was generated with 1% connectivity and random weights drawn from a uniform distribution over (–1,1), then rescaled to spectral radius 0.8. Output feedback connection weights were randomly selected from a uniform distribution over (–1,1). An auxiliary input unit was attached with similar random connections to feed in a constant bias input of size 0.2.

Training and testing. Length of training sequence was 3000, first 1000 steps were discarded to wash out initial transient, echo signals $\mathbf{x}(n)$ were sampled from remaining 2000 steps. Noise $\nu(n)$ of size 1.0e-10 was added on network units during sampling, a trick to increase stability of the trained network, see comments in (*1*). The error to be minimized was $\mathrm{MSE}_{train} = 1/2000 \sum_{n=1001}^{3000} (\tanh^{-1} d(n) - \mathbf{w}_{out}\mathbf{x}(n))^2$. (The $\tanh^{-1}$ was left out in the formula given in the article because it would have required an explanation of the network equations). The weight vector $\mathbf{w}_{out}$ that minimizes this MSE is made from the regression weights of the linear

regression of the $\tanh^{-1} d(n)$ values on the $\mathbf{x}(n)$ values. Any method for computing linear regressions can be used to obtain $\mathbf{w}_{out}$. We employed the pseudoinverse routine `pinv` from Matlab. The obtained training error was $\mathrm{MSE}_{train} \approx 1.2\mathrm{E}\text{-}15$. For testing, 100 independent instantiations of the MGS time series were teacher-forced on the trained network for 2000 steps, then the network was left running freely and the network's continuation was compared to the true continuation after 84 steps to obtain an average normalized root mean square error for the 84 step prediction of $\mathrm{NRMSE}_{84} = \left(\sum_{i=1}^{100} (d(+84) - y(+84))^2 / 100 \sigma^2 \right)^{1/2} \approx 0.000025$ [d(+84) correct continuation, y(+84) network prediction, σ^2 variance of MG signal]. Fig. 2A in the article shows an overlay of a 2500 step free running continuation of the trained network with the continuation of the original MGS series.

In order to test whether the trained network indeed generates a chaotic time series (as opposed to a long limit cycle behavior that might superficially look like a chaotic sequence), the largest Lyapunov exponent of the network-generated output signal was empirically estimated. To this end, the trained network was first run for 1500 time steps in teacher-forced mode, after which it was in state $\mathbf{x}(1500)$. From this state, it was left running freely for 317 steps, and the resulting output sequence $y(1501)$, ..., $y(1817)$ was recorded. In addition, the state $\mathbf{x}(1500)$ was perturbed by a uniform noise vector of size 1e–7, and the network was left running freely starting now from the perturbed state for 317 steps, and the resulting output sequence $y'(1501)$, ..., $y'(1817)$ was recorded. The exponential divergence rate λ between these two sequences was estimated by computing $d_1 = \| [y(1601) \ldots y(1617)] - [y(1601) \ldots y(1617)] \|$, $d_2 = \| [y(1801) \ldots y(1817)] - [y(1801) \ldots y(1817)] \|$, $\lambda = \log(d_2/d_1) / 200$. The subesequence length 17 that goes into the computation of d_1 and d_2 was chosen because it corresponds to roughly one full "loop" of the attractor. Averaging over 100 such trials gave an estimated Lyapunov exponent of 0.0058. An analog empirical estimate of the Lyapunov exponent of the time series obtained from the `dde23` solver yielded a value of 0.0059. Both values agree with estimates for the MG Lyapunov exponent found in the literature, which vary (considerably) around 0.006. Therefore, the network indeed generated a chaotic time series with a degree of chaoticity similar to the original system.

The ESN equalizer

Data preparation. We took the channel model from (*S2*). The channel is modeled by a linear system with memory length 10 followed by a memoryless noisy nonlinearity. The input to the channel is an i.i.d. random sequence $d(n)$ with values from $\{-3, -1, 1, 3\}$. The input-output equation of the linear channel is given by $q(n) = 0.08\, d(n + 2) - 0.12\, d(n + 1) + d(n) + 0.18\, d(n - 1) - 0.1\, d(n - 2) + 0.09\, 1\, d(n - 3) - 0.05\, d(n - 4) + 0.04\, d(n - 5) + 0.03\, d(n - 6) + 0.01\, d(n - 7)$. The noisy nonlinearity is given by $u(n) = q(n) + 0.036\, q(n)^2 - 0.011\, q(n)^3 + \nu(n)$, where $\nu(n)$ is an i.i.d. Gaussian noise with zero mean adjusted in power to yield signal-to-noise ratios ranging from 16 to 32 db.

Network setup. 46-neuron ESN reservoir network weight matrices $\mathbf{W}$ were randomly generated with a connectivity of 20%, and nonzero connection weights drawn from a uniform distribution over (–1, 1). The resulting weight matrix was rescaled to a spectral radius of 0.5. An input neuron was attached with connection weights drawn from a uniform distribution over (– 0.025, 0.025).

The output neuron was a linear neuron here, so the output equation of the network is $y(n) = \mathbf{w}_{out}(\mathbf{x}(n), u(n))$.

Input and teacher signals given to the network. The input given to the ESN equalizer was not the raw corrupted signal $u(n)$ but a shifted version $u(n) + 30$. The teacher output signal was $d(n-2)$, so the learning task consisted in minimizing the square error $(y(n) - d(n-2))^2$. Note that this implies a delay in the equalizer response of 4 time steps, because the signal $u(n)$ depends on d values up to time $n + 2$. The equalizers investigated in (*S2*) had response delays ranging from 8 to 21.

The online learning algorithm. We used the following plain textbook version of the RLS algorithm (*S3*):

Initialization: Create a diagonal auxiliary matrix $\Psi^{-1}(0)$ of size 47 by 47, with large diagonal entries of size 10^{10}. Create an initial all-zero output weight vector $\mathbf{w}_{out}(0)$ of size 47 and an initial all-zero network state $\mathbf{x}(0)$. Define a forgetting rate λ (we used $\lambda = 0.998$).

Input into update cycle n: Previous output weight estimates $\mathbf{w}_{out}(n-1)$ of size 47; network state + input vector $(\mathbf{x}(n), u(n) + 30) =: \mathbf{v}(n)$ of size 47; one-dimensional teacher output $d(n-2) =: d^{+}(n)$; the auxiliary matrix $\Psi^{-1}(n-1)$.

Output after update cycle n: Network output $y(n)$, update $\mathbf{w}_{out}(n)$ of output weight vector, update of auxiliar matrix $\Psi^{-1}(n)$.

Computation steps within one update cycle:

1. $\mathbf{u}(n) = \Psi^{-1}(n-1)\,\mathbf{v}(n)$ [comment: this $\mathbf{u}$ is not related to the input u].
2. $\mathbf{k}(n) = \dfrac{1}{\lambda + \mathbf{v}(n)^{\mathrm{T}}\mathbf{u}(n)}\mathbf{u}(n)$ [comment: $^{\mathrm{T}}$ indicates transpose]
3. $y(n) = \mathbf{w}_{out}(n-1)^{\mathrm{T}}\mathbf{v}(n)$
4. $e(n) = d^{+}(n) - y(n)$
5. $\mathbf{w}_{out}(n) = \mathbf{w}_{out}(n-1) + \mathbf{k}(n)\,e(n)$
6. $\Psi^{-1}(n) = \lambda^{-1}\left(\Psi^{-1}(n-1) - \mathbf{k}(n)\,[\mathbf{v}(n)^{\mathrm{T}}\,\Psi^{-1}(n-1)]\right)$

During the first 100 steps of the 5000-step training run, the RLS update was disabled to allow the reservoir network to wash out initial transients resulting from the arbitrary starting state.

Learning and testing, suite 1 (curve **d** in Fig. 3). We carried out 20 independent learning trials for each signal-to-noise ratio (SNR) 12, 16, 20, 24, 28, 32. A single learning trial consisted of creating a random 46-neuron reservoir and random input connection weights as indicated above, setting the noise $\nu(n)$ to yield a prescribed signal-to-noise ratio, running the RLS algorithm on the random network for 5000 update cycles, freezing the weights and letting the trained equalizer run on a test sequence. The equalized signal $y(n)$ from the test run was converted into a 4-symbol sequence by equidistant thresholding (symbol "1" was chosen if $y(n) < 2$, "2" if $2 \le y(n) < 0$, etc.).

The test run was stopped after the 10th symbol error was encountered or after 10^7 steps, whichever occurred first. The symbol error rate (SER) was estimated by the quotient *number of incorrectly re-converted symbols* / *length of test run*.

Learning and testing, suite 2 (curve **e** in Fig. 3). The reservoir that gave the best performance in the SNR = 32 condition in suite 1 was used in all trials of a second suite of learning trials. The variance found for each SNR in curve **e** from Fig. 3 results from differences in the training and test sequences.

Supporting Material Text

Refined version of the learning method

When the ESN method is used with feedback from the output neuron (as in the Mackey-Glass example), the reservoir network is "forced" by the correct teacher output during the presentation of the training data. When the trained network is later exploited for generating output autonomously, it is receiving its own output signal through the feedback connections instead of the correct signal. Because the self-generated output slightly deviates from the correct one, the dynamics of the reservoir state will be slightly different in training vs. autonomous signal generation. If during training the output neurons could be trained using a reservoir with a dynamics closer to the one encountered at exploitation time, modelling accuracy could be improved. This idea leads to an improved, three-stage version of the training method, which will now be described for the Mackey-Glass prediction task:

1. First stage: an ESN is trained on the Mackey-Glass prediction task exactly as in the basic method, with a teacher time series $d(n)$ obtained from a numerical simulation of the MG equation. This results in a preliminary set of output weights $\mathbf{w}_{out}^0$.
2. Second stage: A new teacher time series $d'(n)$ is computed from $d(n)$ and the weights $\mathbf{w}_{out}^0$, as follows. Put $d'(1) = d(1)$. In order to compute $d'(n)$ for $n > 1$, start network in random state $\mathbf{x}(1)$. To obtain $d'(2)$, write $d(1)$ into the output neuron and update the network once according to $\mathbf{x}(2) = \tanh(\mathbf{W}\mathbf{x}(1) + \mathbf{w}_{fb} d(1))$ and $y(2) = \tanh(\mathbf{w}_{out}^0 (\mathbf{x}(2)))$. Put $d'(2) = y(2)$. Iterate: to obtain $d'(n)$, write $d(n-1)$ into the output neuron, update according to $\mathbf{x}(n) = \tanh(\mathbf{W}\mathbf{x}(n-1) + \mathbf{w}_{fb} d(n-1))$ and $y(n) = \tanh(\mathbf{w}_{out}^0 (\mathbf{x}(n)))$, put $d'(n) = y(n)$. Thus, $d'(n)$ is the sequence of one-step predictions of the network trained in the first stage (except for $d'(1)$).
3. Third stage: Use $d'(n)$ as a new teacher signal and retrain the ESN with $d'(n)$ instead of $d(n)$, obtaining the final set of output weights $\mathbf{w}_{out}^1$.

Because $d'(n)$ incorporates some effects of the autonomous network update dynamics, using it as a teacher in stage 3 brings the network dynamics closer to the autonomous network dynamics in exploitation runs, and thereby yields an altogether more accurate model.

The generation of a synthetic training sequence $d'(n)$ from $d(n)$ can be iterated by generating $d''(n)$ from $d'(n)$ and $\mathbf{w}_{out}^1$ as $d'(n)$ was generated from $d(n)$ and $\mathbf{w}_{out}^0$, etc. However, further such

"relaxation stages" beyond $d'(n)$ did not result in further improvements of model accurary in the Mackey-Glass task (but sometimes did in other chaotic time series prediction tasks not reported here).

When the basic learning method is used to train chaotic attractor predictors, adding noise during sampling time is often helpful to obtain stable network dynamics. This was found to be unnecessary in the refined version of the method.

The improvements afforded by this re-training method can further be enhanced by a standard trick of machine learning: averaging over different models. We demonstrate this on the MGS prediction task (delay 17). Twenty ESN reservoirs of 1000 units each were randomly created, with the same scaling parameters as reported for the basic version. A 3000 step teacher sequence $d(n)$ was computed from the MG equations as described in the methods section. For each of the 20 ESNs, the three-stage refined version of the learning method was carried out independently using $d(n)$, yielding 20 sets of output weights. As in the basic method, network state data from an initial washout period of 1000 steps were discarded from computing the linear regression of the output weights in stages 1 and 3.

For exploitation, the 20 ESN models were combined through averaging their output. Concretely, in each update step n, the 20 networks were updated independently. Their averaged output was fed back into each of the networks. The sequence of averaged outputs was considered as the total model output. The $NRMSE_{84}$ was computed as described above. This entire procedure was repeated 10 times, yielding 10 values of $NRMSE_{84}$, each corresponding to a model of 20 combined 1000-neuron ESNs. The average of the $\log_{10}$'s of the 10 $NRMSE_{84}$'s was –5.09 (stddev 0.25). Compared to the previously achievable accuracy for this prediction task of $\log_{10}(NRMSE_{84}) = -1.7$, this is an improvement of about 3.4 orders of magnitude, or a factor of 2450. Fig. S1 illustrates the prediction error development on long prediction runs.

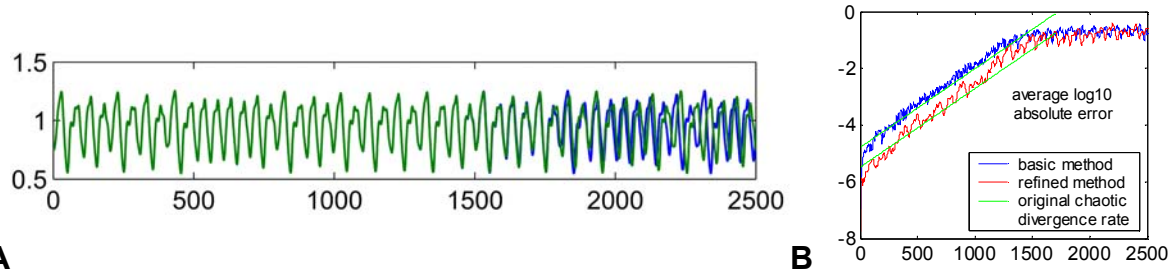


Figure S1: (**A**) Prediction of the MG (delay 17) attractor using the refined ESN method. Blue: correct continuation, green: network prediction. (**B**) Log_{10} of absolute prediction error. Blue: basic method (averaged over 10 networks, 10 predictions each). Red: refined method (one 20-network model, averaged over 20 predictions). Green lines indicate the divergence rate of the original system, that is, the function $\exp(\lambda_1)$, where λ_1 is the largest Lyapunov exponent of the MG system. The x-axis shows network updates (= MG time units) in both plots.

The echo state property

Intuitively, a RNN which is driven by an external signal $u(n)$ has the echo state property if the activations $x_i(n)$ of the RNN neurons are systematic variations of the driver signal $u(n)$. More formally, this means that for each internal unit x_i there exists an "echo function" e_i, such that, if the network has been run for an idefinitely long time in the past, the current state can be written as $x_i(n) = e_i\,(u(n), u(n-1), u(n-2), ...)$. We gave several nontrivial alternative definitions of this condition and algebraic characterizations of which network weight matrices **W** lead to networks having the echo state property (*S1*). For practical purposes it suffices to fix the spectral radius of **W** below unity to ensure the echo state property.

The echo state property is crucial for making the ESN learning method work. Consider some target system of the NMA type (= Nonlinear Moving Average). The current output $d(n)$ of an NMA system is a function of the input history, that is, $d(n) = f(u(n), u(n-1), u(n-2),...)$. Due to the echo state property, the ESN learning approach boils down to approximate the nonlinear system function f by a linear combination of the echo functions, where the linear combination weights are the trained output connection weights: $f \approx \sum_i w_i e_i$. Written out, this becomes $d(n) = f(u(n), u(n-1),...) \approx \sum_i w_i e_i(u(n), u(n-1),...)$. The equalizer system considered in the article is of type NMA.

There are two other major classes of nonlinear dynamical systems, NAR ("Nonlinear Auto-Regressive") and NARMA ("Nonlinear Auto-Regressive Moving Average") systems. NAR systems have system functions of the kind $d(n) = f(d(n-1), d(n-2),...)$ and NARMA systems of the kind $d(n) = f(u(n), u(n-1), u(n-2),... , d(n-1), d(n-2),...)$. The Mackey-Glass attractor is a NAR system. In NAR systems, the next system output is determined by previous output values. This was effected in the MGS study reported in the article through the output feedback connections. In such systems, the teacher output $d(n)$ is approximated through the echo functions by $d(n) = f(d(n-1), d(n-2),...) \approx \sum_i w_i e_i(d(n-1), d(n-2),...)$. The article contained no NARMA example, which however can be learnt in a similar way through ENSs featuring both input neurons and output feedback connections, through $d(n) = f(u(n), u(n-1),..., d(n-1), d(n-2),...) \approx \sum_i w_i e_i(u(n), u(n-1),..., d(n-1), d(n-2),...)$.

Comparison with previous approaches to MGS prediction

The MGS system with delay parameter $\tau = 17$ is a standard benchmark system in research on time series prediction. Almost every available technique for nonlinear system modelling and prediction has been tested on the MGS system. Virtually all previous techniques (except a few which used other versions of RNNs) rely on Takens' theorem (*S4*). This theorem implies that a chaotic system observed in one variable $y(n)$ can in principle be perfectly predicted from a few previous support points, that is, the next value $y(n+1)$ is determined by a few previous output values $y(n)$, $y(n-l)$, $y(n-2l)$, ..., $y(n-kl)$, where l is the distance between support points and k is related to the embedding dimension. Typical values used for k for the MGS are 7 – 10, (7 is the theoretical minimum required). The values taken for l depend on the sampling frequency of the MGS signal. Using this basic approach, the prediction task boils down to estimate from training

data a prediction function f which produces the next value from the support values by $y(n+1) = f(y(n), y(n-l), ..., y(n-kl))$. Methods differ in the mathematical representation of f, which may be expressed for instance by methods of fuzzy splines, feedforward neural networks, radial basis function networks, Volterra expansions, bilinear expansions, support vector machines, or weighted combinations of previously observed subsequences in the training data ("direct methods"). If carried out with care, these approaches invariably yield NRMSE prediction errors for the 84-step prediction in the range of $10^{-1.2}$ to $10^{-1.7}$ (*S5-S11*). In contrast, ESNs do not try to estimate such a prediction function and thus do not ultimately rely on Takens' theorem. The much better precision achieved with ESNs suggest that it is the reliance on Takens' theorem which limited the precision of previous approaches. Specifically, the inevitable numerical inaccuracy entailed by rounding errors implies that the information about the current MGS system state contained in the few support points of traditional techniques is poorer than the massive information that is retained in an ESN "echo" network state about a long previous history (*S12*).

Applying the ESN method to predicting other chaotic attractors

The Lorenz attractor

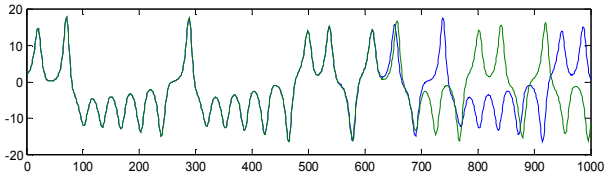
The Lorenz attractor is governed by the three-dimensional ordinary differential equation $dx/dt = p(y - x)$, $dy/dt = -xz + rx - y$, $dz/dt = xy - bz$. We used Lorenz' original parameters $p = 10$, $r = 28$, and $b = 8/3$, for which one obtains a chaotic attractor with a largest Lyapunov exponent of $\lambda_{1\ \mathrm{Lorenz}} \approx 0.906$, which thus exhibits a much stronger chaoticity than the Mackey-Glass system. Training and testing date were prepared by solving the Lorenz equation with the Runge-Kutta (4,5) method, stepsize 0.02 (employing Matlab's `ode45` solver). The stepsize 0.02 was also taken for the ESN update interval, thus one network update covers 0.02 units of original Lorenz time. Following most other learning studies of this attractor, we used only the x-coordinate trajectory for training and testing. The x-coordinate time series thus obtained from the Lorenz equation were rescaled by a factor of 0.01 to obtain series $x'(n)$ to be used for training and testing.

We compared the basic training method, as described in the main text, with the refined method outlined in the SOM.

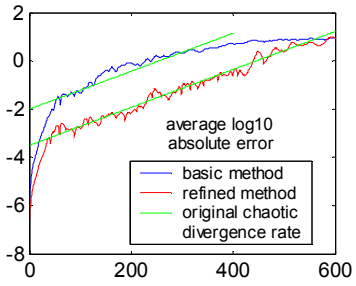
For one training-test run of the original method, a single network of 500 units was created (spectral radius 0.97, connectivity 1/50, input connection weights sampled from uniform distribution over (–1, 1), output feedback connection weights sampled from uniform distribution over (–4,4). It was trained on 6000 update steps on $x'(n)$, with uniform noise sampled from (–1e-8, 1e-8) added to states during updates. Data from the first 1000 updates were discarded. The trained network was tested on 20 independent prediction tasks, each consisting of first running the network through 1000 teacher-forced updates and then letting it run freely to predict the next 600 steps. All error computations were done with data scaled back to the original Lorenz scale. 84-step prediction errors NRMSE_{84} were obtained from the 20 prediction tasks and taken to $\log_{10}$. This train-test procedure was repeated for 10 independently created networks on independently created data. The $\log_{10}\ \mathrm{NRMSE}_{84}$ had an average of –2.33 (stddev 0.54 accross the 10 networks). Fig. S2B shows the average absolute error development.

For the refined method, 20 randomly created networks of 500 units each were trained on 6000 sample points with one relaxation stage. Network scaling parameters were the same as in the basic method. No noise was added to states during sampling. Data from the first 1000 cycles were discarded. The resulting 20-network model was tested by running it through 20 prediction tasks on independent test data. Each prediction task was carried out by first running the model in teacher-forced mode through an initial washout period of 1000 steps and then letting it run freely to predict the continuation for another 600 steps. For the 84-step prediction, an error of $\log_{10}$ $NRMSE_{84} = -4.27$ was obtained, about two orders of magnitude better than with the basic method. Fig. S2A shows a typical prediction, and Fig. S2B shows the average absolute error development on the 600 steps. The prediction reaches its maximal deviation from the correct continuation after about 600 network updates, corresponding to 12 Lorenz time units.

The accuracy relative to the degree of chaos is quite similar to the accuracy seen in the (refined) Mackey-Glass prediction: noticing that the Mackey-Glass predictions reach their maximum average departure from the correct continuation after about 1700 time units (see Fig. S1) we find 12 Lorenz time units * $\lambda_{1\ \mathrm{Lorenz}} = 10.9 \approx 10.2 = 1700$ MG time units * $\lambda_{1\ \mathrm{MackeyGlass}}$.



A



B

Figure S2: (**A**) Prediction of the Lorenz attractor using the refined ESN method. The first coordinate of the attractor is shown. Blue: correct continuation, green: network prediction. (**B**) Log_{10} of absolute prediction error. Blue: basic method (averaged over 10 networks, 20 predictions each), red: refined method. The *x*-axis shows network updates in both plots. Divide by 50 to obtain Lorenz timescale.

A conspicuous feature in Fig. S2B is the initial sharp error rise in the first 40-50 network updates. This results from the fact that the network was teacher-forced by the correct signal before the free-running prediction starts. When the network is decoupled from the forcing teacher at update step 1, it relaxes from a forced mode into its autonomous dynamics, where it settles after about 40-50 updates. A similar relaxation with an initial error rise can be seen in Fig. S1B, although it is less pronounced there.

The Lorenz attractor has been used often to demonstrate time series prediction methods. Unfortunately, prediction studies using the Lorenz attractor vary in important characteristics (number of training points, sampling rates, prediction horizons, error criteria, dimensions used for training), which makes a comparison difficult. The best Lorenz prediction study (*S13*) known to the authors manage to keep track of the Lorenz time series through 4 "switches" of the attractor from one lobe to the next, while our ESN model (refined version) stays with the true continuation for 5-11 such switches (mean 7.0 on 20 visually inspected predictions, stddev 1.6). Considering the fact that roughly such switching events occur after 2 Lorenz time units on average, in order to

stay with the correct continuation for one more such switch one needs an additional accuracy factor of $\exp(2*\lambda_{1\ \mathrm{Lorenz}}) \approx 6$. A prediction method *A* that stays with the correct continuation for 3 more lobe switches than another prediction method *B* would therefore have to be about $6^3 \approx 1e2.2$ times more accurate (in terms of absolute distance between predicted and correct trajectories).

The Mackey-Glass attractor with delay 30

Besides the MG attractor with a delay of 17, the other widely investigated version of the MG attractor is defined by putting the delay to 30. This attractor still is only mildly chaotic ($\lambda_{max} \approx 0.007$), but the resulting time series is phenomenologically much richer than for $\tau = 17$ and therefore more difficult to learn. We applied the basic and the refined version of ESN training to this task.

Training and testing data were prepared by simulating the attractor with a stepsize of 1.0, using the `dde23` solver of Matlab with absolute accuracy set to 1e-16. The resulting time series were subsampled by 6, such that one increment in the resulting time series $d(n)$ corresponds to 6 time units of the original MG equation, a common setting in MG modeling studies. These data $d(n)$ were brought into a range suitable for training by putting $d'(n) = (0.3 * \tanh(d(n) - 1)) + 0.2$.

For testing the basic learning method, we created 10 random ESNs (500 neurons each, connectivity 1/50, spectral radius 0.9) with a single input and a single output neuron. Input connection weights were sampled from a uniform distribution over (–0.2, 0.2), and output feedback weights from a uniform distribution over (–1, 1). The networks were individually trained with the basic method on a 3000-step sequence $d'(n)$ with a constant "bias" input of size 1.0. While the network was driven by the teacher data, uniform noise $\nu(n)$ sampled from (–0.0008, 0.0008) was added on the network states. Data from the first 1000 steps were discarded before computing the output weights. Figure S3B shows the average absolute prediction error for predictions lasting 150 network updates. For all test error computations, the network outputs were first retransformed back to the original MG scaling and compared with the originally scaled MG signals $d(n)$.

For testing the refined method, ten independently created ESNs (700 neurons each, one input neuron, one output neuron, spectral radius 0.85, connectivity 1/70) were employed in the refined learning method. Input and output feedback connection weights were randomly assigned like in the basic method. The model was trained by teacher-forcing it through 3000 data points, of which the first 1000 were discarded before computing the regression of the output weights on the network states. Figure S3B shows the average absolute prediction error on independent test data. A typical prediction is shown in Fig. S3A. Both graphics demonstrate that the model loses track of the original after about 120 iterations, corresponding to 720 MG time units.

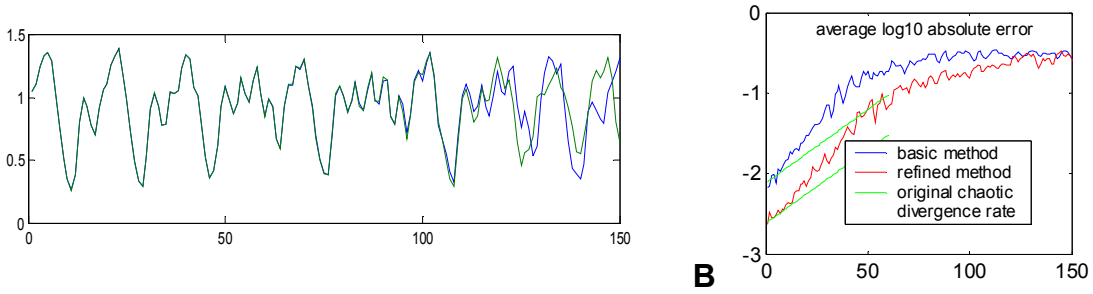


Figure S3: (**A**) A sample prediction with the refined ESN model. Blue: true continuation, green: model prediction. (**B**) Average absolute prediction error of ESN models for iterated prediction. For the basic method, the average is taken over 10 networks and 20 predictions each, for the refined method over 50 predictions of a single model made from 10 combined ESNs. x-axis corresponds to network updates in both plots. Multiply by 6 to obtain MG time units.

From a perspective of nonlinear dynamics, the delay = 30 case of the MG attractor is more difficult to model than the delay = 17 case not because of different degrees of chaos (they are almost the same) but because of a higher embedding dimension of the delay-30 attractor (*S14*), namely, 4 vs 3. Intuitively this means that the trajectory of the delay-30 attractor "lives" in a 4-dimensional space, as opposed to a 3-dimensional space for the delay-17 case. This has dramatic effects on learnability: if training information from 2000 points is dispersed in a 4-dimensional space as opposed to a 3-dimensional space, it covers state space volume much more sparsely. In order to obtain the same covering density as in the delay-17 case, $2000^{4/3} \approx 25{,}000$ training points would have been needed to ensure a similar model precision. In addition, the geometrical shape of the delay-30 attractor is much more involved than the delay-17 attractor (not discussed here; intuitively, it is much more "curly"), which again would require a relatively higher density of training points to obtain a similar model precision as in the delay-17 case. Both effects taken together render the delay-30 MG attractor a formidable modeling task.

The best approach to modeling this attractor we are aware of (*S15*) obtains a root mean square error for a prediction of 120 MG time units of 0.04, which corresponds to a $\log_{10}$ NRMSE$_{120}$ of –0.15. Our corresponding figures for the $\log_{10}$ NRMSE$_{120}$ are –0.84 on average for the single nets trained with the basic method (stddev over the individual 10 networks' $\log_{10}$ NRMSE$_{120}$: 0.083) and –1.42 for the refined method.

The Laser time series

One of the time series used in the 1994 time series prediction competition (*S16*) organized by the Santa Fe Institute was an empirical time series obtained from measuring a pulsing laser. This data set has been tackled with virtually every time series prediction method available. Figure S4A shows the data as they were published for the competition; the task was to predict the next 100 steps.

Using the refined method, we trained a collective of 20 ESNs with 300 neurons each on the original data set. The 200-step prediction obtained is shown in Fig. S4B. This prediction matches

the first high spikes very well, accurately catches the breakdown event, but is out of phase in the ensuing buildup of the oscillation. Very similar outcomes have been observed by a number of other researchers; no-one could catch the phase in the final buildup section correctly. The reasons for this have been elucidated (*S17*), as follows. The training dataset contains exactly one instance of a breakdown event like the one in the prediction interval (red box in Fig. S4A). The antecedent oscillations of this event in the training data match the oscillations right at the end of the training data and the first few steps into the prediction interval extremely closely. Figure S4C shows an overlay of the event in the training data with the event of the true continuation to illustrate this finding. Assuming that the physical laser is a deterministic system, the best prediction is therefore just to replicate the continuation from the similar section in the training data. This is what the ESN model does, as illustrated in Fig. S4D. Thus, the ESN model has "found out" that it faces essentially a pattern matching problem.

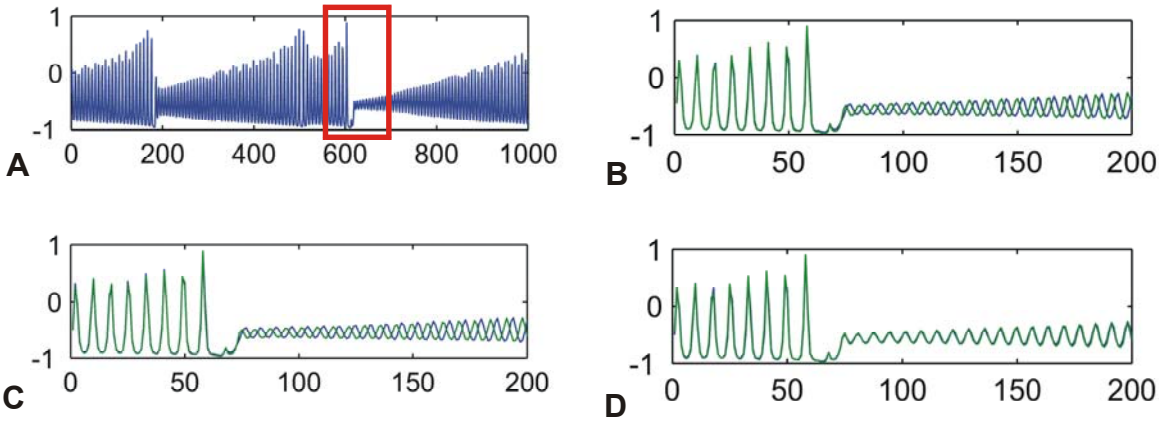


Figure S4: Predicting the Laser time series. (**A**) Training data. (**B**) ESN prediction (green) vs. true continuation (blue) after time 1000. (**C**) Situation in training data (green) [= data from red box in **A**] vs. true continuation (blue) after time 1000. (**D**) Situation in training data [= data from red box in **A**] (blue) vs. ESN prediction (green) after time 1000.

This problem exhibits specific difficulties not present in the other examples. First, the original data carry substantial numerical roundoff noise (they are measured with 8 bit precision), which has prompted most researchers to preprocess the dataset by interpolation/smoothing. We used the raw data. Second, the time series has two very different timescales: a fast oscillation and a slow buildup-breakdown cycle. Third, the breakdown event that has to be predicted occurs only once in the training data. All of these difficulties are not serious if one treats the problem as a pattern matching task right away and solves it by detecting the fact that the immediate antecedent of the prediction interval occurs already earlier in the training data, and re-using the original continuation. This is what many successful solutions do. However, if this valuable insight is not made or not used, the three difficulties become formidable. From a machine learning perspective, especially the third difficulty is noteworthy: we face here an instance of the "single shot learning" problem, that is, a model has to capture some phenomenon from a single presentation during training.

Supporting Online Material References

S1. H. Jaeger, "The echo state approach to analysing and training recurrent neural networks" (GMD-Report 148, German National Research Institute for Computer Science 2001). ftp://borneo.gmd.de/pub/indy/publications_herbert/EchoStatesTechRep.pdf

S2. V. J. Mathews, J. Lee, in *Advanced Signal Processing: Algorithms, Architectures, and Implementations V (Proc. SPIE Vol. 2296)*, (SPIE, San Diego, CA, 1994), pp. 317-327.

S3. B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications* (Wiley, New York 1998).

S4. F. Takens, in *Dynamical Systems and Turbulence*, D.A. Rand, L.-S. Young, Eds. (Springer Lecture Notes in Mathematics **898**, 1991), pp. 366-381.

S5. J. McNames, J. A. K. Suykens, J. Vandewalle, *Int. J. Bifurcation Chaos* **9**, 1485 (1999).

S6. J. Vesanto, in *Proc. WSOM'97* (1997). http://www.cis.hut.fi/projects/monitor/publications/papers/wsom97.ps.zip

S7. L. Chudy, I. Farkas, *Neural Network World* **8**, 481 (1998).

S8. H. Bersini, M. Birattari, G. Bontempi, in *Proc. IEEE World Congr. on Computational Intelligence* (IJCNN '98) pp. 2102-2106 (1997). ftp://iridia.ulb.ac.be/pub/lazy/papers/IridiaTr1997-13_2.ps.gz.

S9. T. M. Martinetz, S. G. Berkovich, K. J. Schulten, *IEEE Trans. Neural Networks* **4**, 558 (1993).

S10. Yao, X. & Liu. Y. *IEEE Trans. Neural Networks* **8**, 694 (1997).

S11. F. Gers, D. Eck, J. F. Schmidhuber, "Applying LSTM to time series predictable through time-window approaches" (IDSIA-IDSIA-22-00, 2000). http://www.idsia.ch/~felix/Publications.html.

S12. H. Jaeger, "Short term memory in echo state networks" (GMD-Report 152, German National Research Institute for Computer Science 2002). ftp://borneo.gmd.de/pub/indy/publications_herbert/STMEchoStatesTechRep.pdf

S13. J. McNames, PhD thesis, Stanford University 1999. http://www.ece.pdx.edu/~mcnames/Publications/Dissertation.pdf

S14. J. D. Farmer, *Physica 4D* (1982).

S15. L.A. Feldkamp, D. V. Prokhorov, C. F. Eagen, F. Yuan, in *Nonlinear Modeling: Advanced Black-Box Techniques*, J.A.K. Suykens, J. Vandewalle, Eds., (Kluwer 1998), pp. 29-54. http://mywebpages.comcast.net/dvp/bpaper.pdf

S16. A. S. Weigend, N. A. Gershenfeld (Eds.), *Time Series Prediction: Forecasting the Future and Understanding the Past* (Addison-Wesley, Reading, Mass., 1992).

S17. J. Kohlmorgen, K.-R. Müller, *Neural Processing Letters* **7**(1), 43 (1998). http://www.first.gmd.de/persons/Kohlmorgen.Jens/publications/datasetA.pdf